User Documentation

Contents

Section 1 - Purpose of this document

This document aims first to give anyone interested an insight into how the system came to be developed.

Secondly, it enables someone with an Atom to explore the facilities available with the system. A basic knowledge of the Atom is assumed, but not a knowledge of BASIC or any other language. It is, however, written for an adult reader. A separate guide for children is being prepared.

Thirdly, it will serve (for the time being) as a reference manual.

Section 2 - Background to the system

This system has arisen as a result of difficulties I have found while teaching BASIC to a variety of newcomers to computing in both informal computer literacy classes and formal GCE and adult education classes. Those reading this who are in a similar position will, I hope, sympathise with these problems, some of which are :-

1) BASIC is derived from a mathematical problem-solving language (FORTRAN), and it often betrays it's origins. Even worse, I think, is that it is reminiscent of Algebra, probably the most hated mathematical subject at schools (which seems to affect adults more than children)

2) There is too little you can do in BASIC's immediate mode; by that I mean without actually writing a program - simply keying in command for immediate execution. (I do realise that most other languages do not even have an immediate mode). All you can usually do is

```
PRINT "HELLO"
or  PRINT 2+2
```

neither of which is very interesting or impressive. To do more, involves learning about relatively unfamiliar concepts like line numbers, and commands like LIST and RUN. Even then you only have

```
1Ø PRINT "HELLO"
2Ø GO TO 1Ø
```

which leads on to complications like the ESC key to stop the thing. To do a more controlled program like printing HELLO 10 times needs so much more; ideas like variables and either FOR/NEXT loops or IF/THEN/GOTO's with peculiarities like LET X=X+1. BASIC needs so much explaining, for little end result.

3) The error checking and error messages of all common BASICs are a disgrace. This is more a criticism of implementation than of design. It creates so many problems when you are allowed to enter a line of program of any old rubbish; most BASICs only check the syntax in detail when the program is run. And then they give

```
?SN ERROR IN LINE xxx
```

Not a hint of where, or why the error happened. I think it shows a lack of correct priorities that as BASIC interpreters have grown from 4 to 8 to 16k that error diagnosis has not improved. I would willingly swap logs and trig. functions for real error messages.

The most frustrating thing is that the interpreter probably knows where and why the error occurred, but it is not telling. Sinclair deserve much credit for their ZX series interpreters, which do tell you where (if not why).

4) It is very difficult to teach structured programming in BASIC. This has been written on at length elsewhere, so I will say no more here.

About a year ago, I was introduced to the Logo programming language developed at MIT in the States over the past 10 years or so. At first glance, this language system seemed to answer all my problems.

It is based on turtle graphics and the Lisp language, and is about as far from a mathematical problem solving language as you can get. The original "turtle" was a box on wheels that presumably looked something like it's namesake. It had a pen attached to it, and could be programmed to trundle across the floor, leaving a trail. You first had to feed in instructions for it to move so many feet forwar or back; to turn to the left or right so many degrees; lift the pen up and put it down. Then on pressing GO it would obey your program in a very computer-like obedience.

Logo does much the same, except the turtle draws on a high resolution video screen from instructions typed in at a keyboard. It is being used with much success in schools from Primary level upwards, as both an introduction to computers and as a tutor in geometry.

To answer my second complaint about BASIC, it has superb error diagnostics. Add to that the fact that it has a very rich immediate mode, which means that writing programs is postponed until they are actually needed. When they are written, they are really procedures with parameters which can be called up when required; Logo is a totally structured language where small procedures can be nested inside one another. These user-written procedures are thought of simply as extensions to the language.

I hope that introduction to Logo has given an appreciation of it's qualities. However there is much more to it than that, wherein lies the problem. The original Logo was written, I believe, for a PDP11. A version is being developed for a 48k Apple with a disk. That is fine, except that Logo is a <u>one-user</u> system; it is a personal environment for one person to explore and learn from their own mistakes, at their own speed with minimum supervision. Until the day comes when a classroom full of Apples connected in a network becomes part of every school, there is little chance of Logo taking off.

So, this system, which has been named "Turtle talk", is an attempt to provide a Logo-like environment on an Acorn Atom. It requires a fully expanded Atom with floating point chip, but even so the cost is only around £250. The BBC micro would enable a much better version to be implemented at much the same cost.

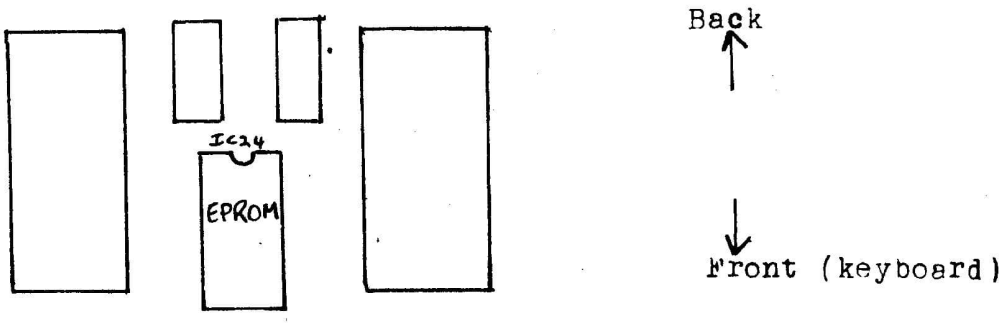
It's aim is to get over the concepts of structured programming to a newcomer in a gradual, interesting way with the minimum need for supervision and tuition. An understanding of some geometric principles comes as a by-product!

For further details on Logo, and turtles in education, read the book "Mindstorms, Children Computers and powerful ideas" by Seymour Papert, published The Harvester Press, 1980.

Section 3 - Installing and loading-Model A

The program comes as a cassette and an EPROM chip. The cassette contain the BASIC program, and the chip contains tables, error messages, some machine code and a couple of BASIC subroutines.

Install the EPROM first. Ensure computer is switched off, and remove the printed circuit board from the case by removing the screws. Turn the board over, and identify the 24-pin socket labelled IC24. It is in the centre of the board. Carefully insert the EPROM into the socket with the small notch in it's case towards the rear of the Atom. Use no force - simply gentle persuasion!



Before relacing the screws, connect the power supply & check that the EPROM is installed correctly. Type in

P.$#A644

and the message "EPROM CORRECTLY INSTALLED" should appear in inverse video.

Re-assemble the Atom, and prepare to load the cassette. To help you get the right volume setting, there is an initial file on side one of the cassette before the program itself. Type

*LOAD "" 8000 (note no spaces between the "")

and play the cassette. If your volume setting is correct, then the top half of the screen should fill with T's, and the cursor re-appear.

If that is correct, rewind the cassette, and type in

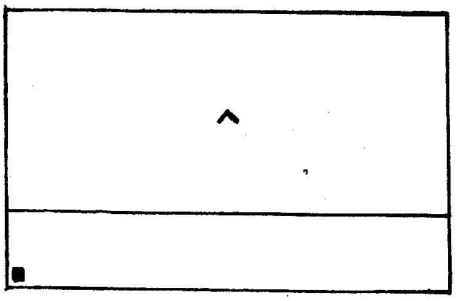
LOAD "TT"

When it is successfully loaded, it is suggested you save it immediately onto a cassette of your own, befor RUNning.

Section 4 Exploring the system

4.1 Introduction

Assuming you have just loaded the program and typed RUN, you should be faced with a (graphics) screen divided into 2 sections. In the upper section is a small arrow head; this is the tip of the "pen" by which you can draw anywhere within that section. In the lower section is a square cursor; this is the text section and holds 4 lines of text.



4.2 Drawing lines

Key in

FORWARD 40 and press RETURN (FORWARD could be abbreviated as FD)

This is the command to move the pen forward 40 units in the direction in which it is facing, so a line a couple of inches or so in length will be drawn towards the top edge. The screen is 159 units high in total, so this took you about half-way to the top edge.

Now type in the command to turn the pen through 90 degrees to the right (for the remainder of this section, the need to press RETURN will not normally be mentioned)

RIGHT 90 (or RT)

Now do another FD 40. Since the screen is 256 units across, this will take you approximately one third of the way to the right-hand edge.

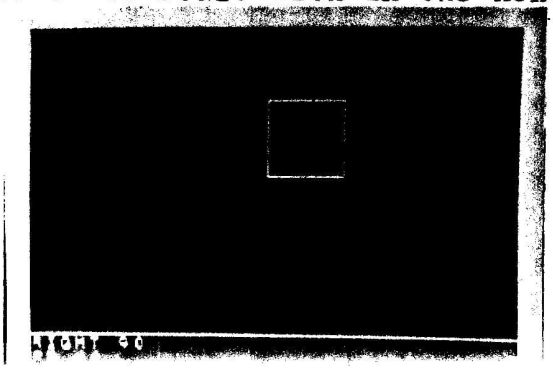
4.3 Drawing a square

You can complete a square by drawing the remaining 2 sides in one command

RT90 FD40 RT90 FD40 RT90

Note that spaces can be inserted or left out as you wish; there is only one place where spaces are needed, and that is between 2 (alphabetic) words

You should have a square on the screen, with the pen back in it's original position. On the other hand, perhaps you have had a bleep and an error message. These are dealt with in the next section.



4.4 Error messages

There are two types of error messages that can be detected. First, those detected immediately after a line has been typed, where the system thoroughly checks the syntax of a command. If you have not made an error, type in

FD 5Ø RUBBISH FD 5Ø

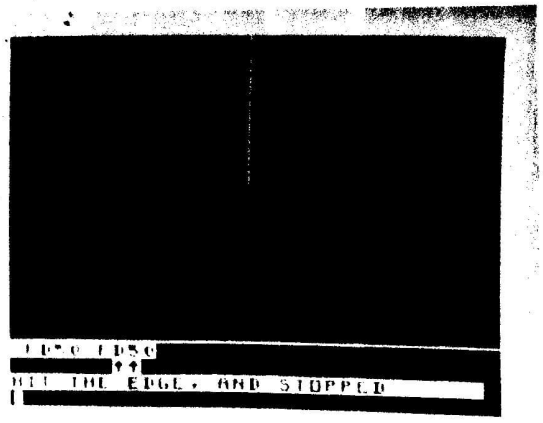
and you will get an appropriate message concerning the word RUBBISH. Most error messages are self-explanatory, and are listed in Appendix B. Note that it will be necessary to re-type the line completely to correct the error. Also note that all messages from the system are displayed in inverse video.

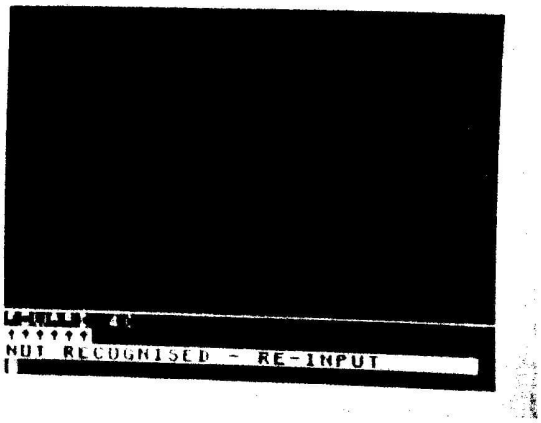
The second type of errors are those which can only be detected while the system is drawing (run-time errors). For example, type in

FD 5Ø FD 5Ø .

and this will be accepted (indeed anything up to FD 36Ø will be accepted) however since the second FD50 would have taken the pen above the top of the screen, you will get an error message, and the pen will stop at the top edge.

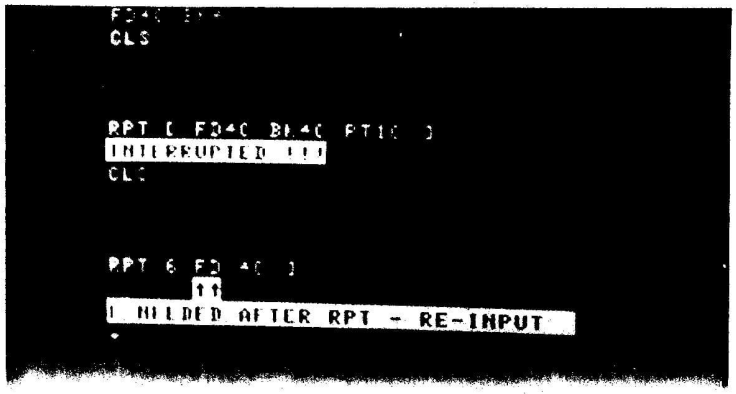
While mentioning errors, note that the DELETE key works as normal in deleteing the last character keyed in.





4.5 Full-screen text

Normally, only 3 or 4 lines of text can be seen at the bottom of the screen. However, the last 15 lines of text are stored by the system, and can be seen by pressing RETURN with the cursor in the left-most position. Another RETURN then gets you back to the graphics screen, or you can type in your next command from the full-text screen.



4.6 The Repeat command

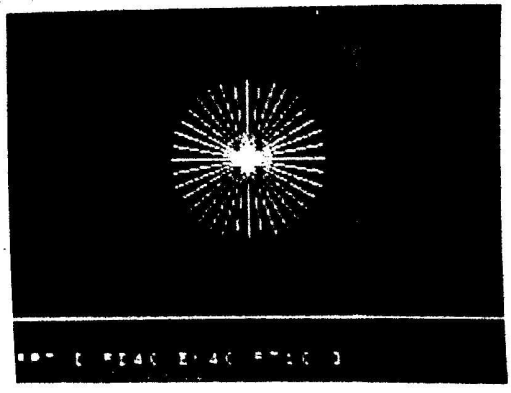
First clear the screen by keying

CLEARSCREEN (or CLS)

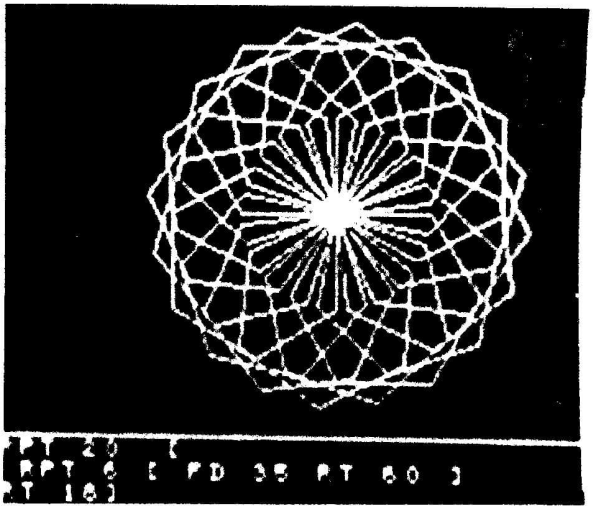
Drawing that square back in section 4.3 was a rather repetitive task there were 4 lots of "FD40 RT90" sequences. Fortunately Turtle Talk (and most programming languages) give you methods of repeating sequences of commands as many times as you require. In Turtle Talk this is the REPEAT command. Type in

REPEAT 4 [FD 40 RT 90] (or RPT for REPEAT)

and everything inside the square brackets will be repeated 4 times. This is a great time-saver, and can give some spectacular patterns easily. Try these



RPT [FD40 BK40 RT10]



```
RPT 20 [
      RPT 6 [ FD35 RT60 ]
RT18 ]
```

Note a couple of things from these examples :-

a) If you don't give a number after the word RPT then the repeat is done indefinitely. To break out of this never-ending loop, press the BREPT key in the bottom right hand corner of the keyboard. This key acts as an interrupt key. Do <u>not</u> use the ESC key for this purpose.

b) A RPT may be split over several input lines provided you split it at a sensible point; i.e. between 2 commands. The RPT will not be done until the final] is input.

4.7 Defining Procedures

The next step is to increase the "vocabulary" of the system by definin new words. Going back to the original square of sides length 40, if you wanted to experiment with squares of that size, it would be good to give the system the definition of that square, and call it up when you needed it. So, add the name SQ to the system as follows. Type

DEFINE SQ (or DEF or TO as alternati to DEFINE; note there has be a space between the 2 words)

and you will be asked for the definition of SQ

RPT 4 [FD40 RT90] END

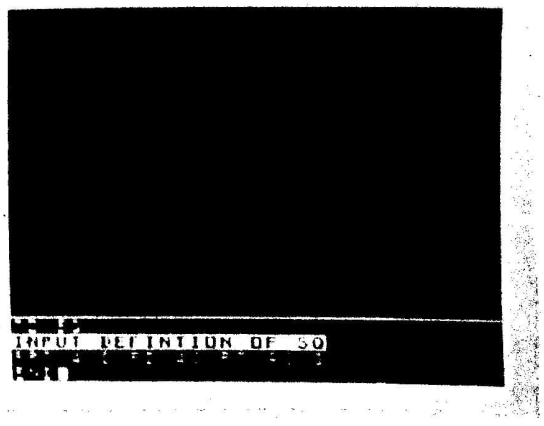
Note the extra word END needed to signify the end of the definition. This is needed because a definition can be spread over many lines. Assuming it was accepted, the word SQ will now be recognised as a command. Type in

CLS SQ

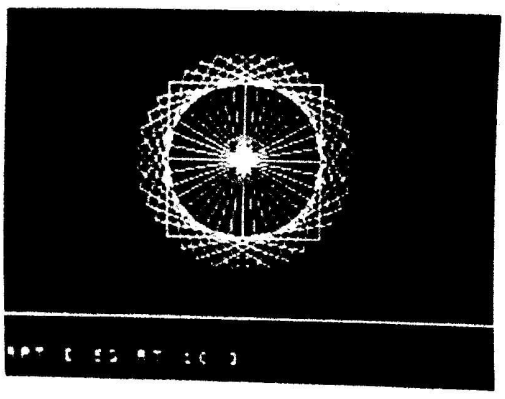
and you should get the square. You could have called it anything instead of SQ provided it was alphabetic, and was not the same as a "reserved wor (i.e. a word already known to the system - see Appendix A for a full list

These definitions are known as "procedures" in Logo, which is not a friendly word, but one that it is worth getting used to.

Once you have SQ defined, it can be put inside a RPT, or called from inside another defintion.



Defining SQ



RPT [SQ RT 10]

You can now go on and define other procedures for triangles, hexagons and so on. However you will eventually run out of memory - there is not much space unless you have external expansion RAM fitted. Once you have r out of space the only answer is to delete some.

DELETE name	(or DEL) will delete a procedure
LIST	will give you the names of all you have defined
LIST name	will give you the definiti of this name

4.8 Parameters for procedures

Procedures like SQ to draw a square of sides length 40 are not very flexible. A big improvement would be to give it a parameter so that it draws squares of any size. A parameter in this sense is a name which stands for a value which you will give to the proc. when you actually execute it.

So type in this line, which will redefine the name SQ

```
DEF SQ LEN
```

here LEN is the parameter name. There must be a space between SQ and L otherwise the system will think you are defining a proc called SQLEN.

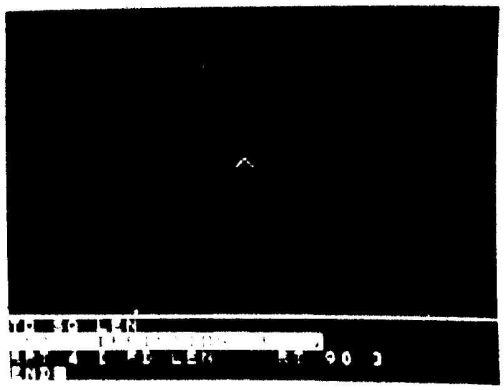
Then type in the definiton for a proc. which will draw a square of any size, since the length of the side is being called LEN.

```
RPT 4 [ FD LEN RT 90 ]
END
```

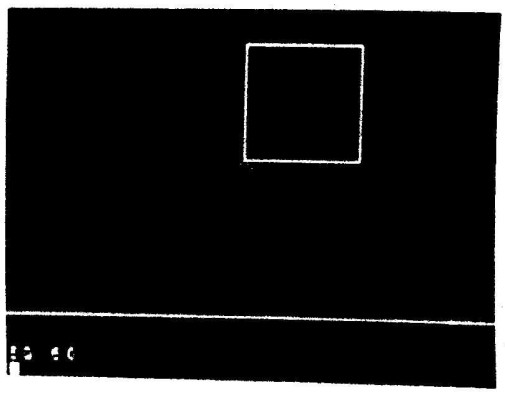
Now to draw a square of sides length 70, type

```
SQ 70
```

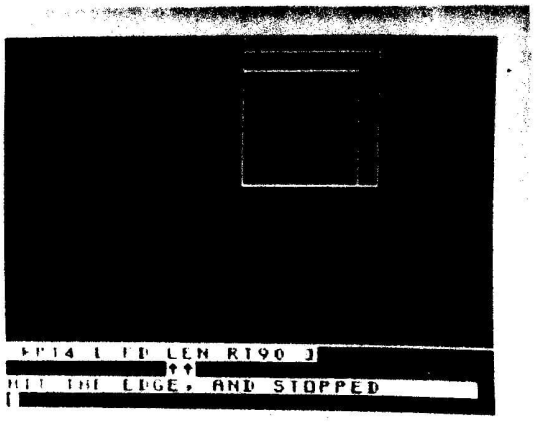
and so on for squares of any other size.



Defining SQ with parameter LEN



SQ 60



Can't do SQ 80!

Rules for parameter names are identical to those for proc. names given in 4.7. However one interesting thing to note is that the name LEN is entirely local to the proc. definition; two or more procs can use the parameter name LEN without any confusion.

4.8 (cont'd)

Procedures can have up to 3 parameters; as an example of a proc. with 2 parameters, here is one to draw any regular polygon of any size.

```
DEF POLY SIDES LEN

RPT SIDES [ FD LEN  RT 360/SIDES ]
END
```

The RT turn is given as 360 divided by the number of sides the poly. has. Expressions like this may be given instead of a simple number, both in proc definitions and normal commands. For example

FD 200/4 is the same as FD 50

Expressions can be of any length, provided they follow these rules

a) Only 4 operators + - * / with no unary - (i.e. FD-6 is an error)
b) No brackets; expression is evaluated strictly left to right
c) Division by zero and any expression with a result >±32767 are error
d) Integer arithmetic is done. I.e. results are rounded to the nearest whole number

Now experiment with POLY. Draw a hexagon (POLY 6 45); a deca-octagon (POLY 18 10) which is nearly a circle. Note a uni-decagon (POLY 11 15) will give slight rounding errors since 360/11 is rounded from 32.727 to 33. Try a nullagon (POLY 0 25); this will give nothing since a RPT done zero times is ignored.

4.9 Saving and loading

The "library" of procedures you have defined may be saved to cassette tape, and reloaded another time. To do a save, type

SAVE

The message RECORD TAPE will appear; this has the same meaning as when it appears when saving a program since Turtle Talk uses the routines within the Atom Operating system. So set the cassette deck to record, and press any key. The message COMPLETED will appear when saved.

To retrieve it, type

LOAD

and on PLAY TAPE set the cassette deck to play, and press any key. This will load from the tape to replace all procedures currently defined.

Since you are using the Atom routines, any errors it finds are liable to result in the program terminating. It will be necessary to type RUN to get it going again.

4.10 Odds and Ends

Here are a few facilities not mentioned so far.

The pen can be moved both BACK (BK) as well as FORWARD (FD). It can be turned both to the LEFT(LT) as well as the RIGHT (RT).

The command HOME will bring the pen back to the centre of the screen facing upwards as CLS does, but it does not clear the screen.

If a PENUP (PU) command is executed, the any further movements of the pen will not leave a trace, until a PENDOWN (PD) or a CLS is executed. (A CLS does an automatic PENDOWN; a HOME does not)

The TURTLE TALK drawing when the program is run is done by using PENUP and PENDOWN, together with left and right turns. Another use of pen up would be to start drawing a shape in a place other than the centre of the screen.

The pen can be put in two other modes- PENERASE which will not leave a trace and also erase any lines exactly traced over again. And PENREV (PR) which will leave a trace, except where it traces over a line, which will be erased.

The command NOEDGE will allow the pen to move outside the screen area without giving an error message, as is the normal case. EDGE will return to the normal situation. These 2 commands must be given as the only items on an input line, and cannot be made part of a procedure definition.

The command BYE will stop the program running, re-setting any system variables altered by Turtle Talk. Pressing BREAK will have much the same effect. You will lose any procedures, unless you have saved them to tape. The ESC key is disabled while the cursor and pen are flashing on the screen. However it is still active while the system is drawing; if used it will stop the program without re-setting any variables, so is best avoided.

Turtle Talk can draw "random" patterns; or to be more precise, the random number generator RND can be put in any expression. RND will give a random value from zero to nine inclusive. Try

```
RPT [ RPT 5 [ RT RND*60   FD RND*5 ] HOME]
```

It would be best to do a NOEDGE before trying this, since this pattern will occasionally go off the edge.

RPTs may be "nested" like this, with one inside another. Procedure calls can also be nested; with proc. A calling proc. B. Then when B is done, control returns to A. The values of parameters can be passed from one to another (see 4.12 for an example of this). One point to bear in mind is that the system has a limited size area called the stack to hold information while executing RPTs and proc. calls. If you nest them too deeply, you will get an error message. It is very unlikely this will be a problem in normal running. The next topic, recursive procedures, is a different matter though.

4.11 Recursive procedure calls

A recursive procedure is one which calls itself. This is perfectly legal, and occasionally useful. An example would be a proc. to draw a spiral. To give you an idea of the method, enter this

```
FD9 RT55 FD9 RT5Ø FD9 RT45 FD9 RT41
FD9 RT37 FD9 RT33 FD9 RT3Ø FD9 RT27
```

This is a sequence of FD9 followed by RT turns which start at 55 and decrease by about 10% each time. Even if you eliminate all spaces, it will take two lines to input it.

This gives the beginnings of a spiral. If can alternatively be done by a recursive proc. definition. Define

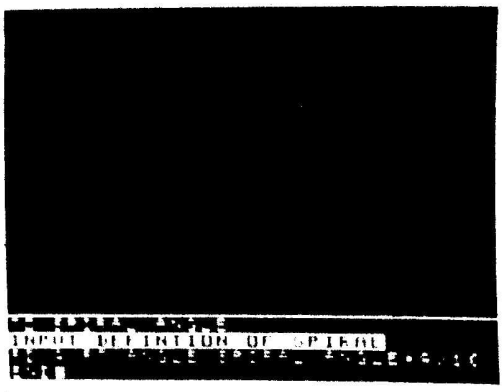
```
DEF SPIRAL ANGLE
```

where ANGLE is the angle at which the pen is to turn after each FD9. It is to decrease to 9/10 of it's value each time. So the definition is

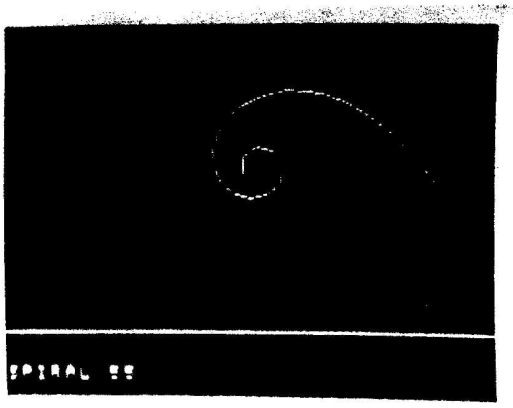
```
FD9  RT ANGLE  SPIRAL ANGLE*9/1Ø
END
```

Thus the procedure "loops back" on itself. If it started off by a call of SPIRAL 55, it does the FD9 RT55, and then calls SPIRAL 55*9/10 (or 50) and so on.

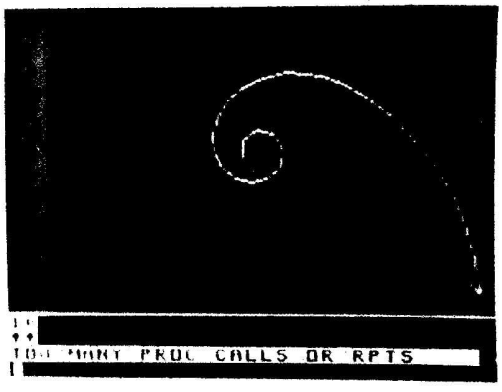
Eventually the system will run out of stack space after about 30 calls. However, you have the additional advantage of being able to start the spiral off with different values for ANGLE.



Input the definition



Here it goes...

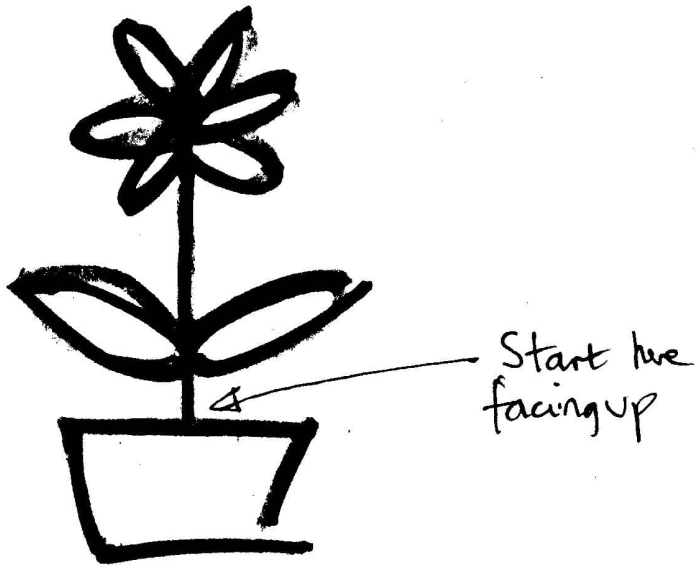


But eventually runs out of stack space.

4.12 Making full use of procedures

The concept of procedures (with parameters) is a very powerful one; a concept which is sadly lacking from many computer languages. An example of a procedural approach using Turtle Talk is given next. This example is based on one given in Mindstorms (see end of Section 2).

The aim is to draw a pot plant like



The problem must be broken down into manageable size pieces. In this case that would be the pot, the stem, the 2 leaves and the flower head. Next, recognise economies that can be made - the leaf and a petal are much the same shape, just a different size.

Having split up the problem, there are two approches to actually programming it; either from the "top down" or the "bottom up". Either approach can be used in turtle talk. First, then, I will explain the top down; starting by defining the whole thing, and then it's constituent parts.

So, type in

```
TO POTPLANT              ( TO is an alternative to DE
                         ( use whichever you prefer

POT
FD 1Ø
LEAVES
FD 4Ø
FLOWER
END
```

Now, what about the pot? Before defining it, you can experiment moving the pen around from the home position to get a good pot. The only real requirement is to start and end in the same position, facing the same way. It makes things a lot easier if, wherever possible, all procedures do this. This will make a reasonable pot.

```
RT 9Ø
FD 2Ø RT 12Ø FD 3Ø RT 6Ø
FD 1Ø RT 6Ø FD 3Ø RT 12Ø FD 2Ø
LT 9Ø
```

So, define that (or your improved version) as POT. Remember that pressing RETURN with the cursor in the left-most position will give you a reminder of the last 15 lines displayed, which can be very useful.

OK, now for the leaves, and remember that a leaf is really a large petal. So how about

```
TO LEAVES

RT 60      PETAL 40
LT 120     PETAL 40
RT 60  END
```

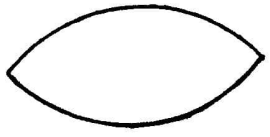
That should give one "petal" of length 40 to the right of the stem, & one to the left, with the turtle ending up where it started.

What about the flower? It is going to be a repeat of, say, 8 petals of size 30.

```
TO FLOWER

RPT 8 [ PETAL 30  RT 360/8 ]    END
```

Note again a 360 degree turn is involved. Now for a petal. This is really 2 quarter circles joined together with turns at the end.



So a QCIRCLE procedure would be a good idea, which draws a quarter circle of any size. Try this out, before you define it

```
RPT 5 [ FD 5 RT 18 ]
```

Note we are only turning the turtle through 90 degrees in this (5 times 18). That seems OK, so define

```
TO QCIRCLE SIZE

RPT 5 [ FD SIZE/5  RT 18 ]      END
```

and, of course, the FD step is SIZE/5 for each of the 5 steps. Back to petal. It will be

```
TO PETAL SIZE

RPT 2 [ QCIRCLE SIZE    RT ___ ]      END
```

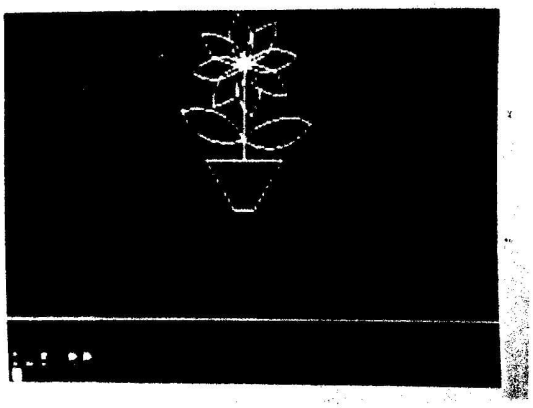
What will be the right turn at the end of each quarter circle? I will leave it for you to fill in. Just remember that you want to do a full 360 degree turn in total.

That is it - apart from a bug in the leaves area. Try it out, and correct the bug.

The other approach to the "problem" is to work from the bottom up; starting with the petal and quarter circle, defining them and getting them correct. Then the flower, then the stem and leaves and the pot, ending up with defining the potplant from it's constituent parts. (The example in Mindstorms takes this approch). Perhaps this approach does make de-bugging and testing easier, but the top-down method certainly (to me) mirrors more the way I think.

Turtle Talk allows you to tackle it either way, or even a combination of the two if you like.

As a further development, how about turning POTPLANT into a variable procedure, with a parameter to draw pot plants of any size? You could draw half a dozen different sized plants, and end up with a green-house!



4.13 Conclusion

That concludes this guided tour to the facilities. A summary of the commands is give in Appendix A, and the enclosed summary card. If you have any suggestions for improvement, please let me know.

~~5 Edinburgh Road~~
~~Henwell~~
~~London W7 3QT~~
23 Dalford Court
Hollinswood
Telford Salop.

Command	Alternatives	Explanation
HOME		Put pen in home position - at co-ordinates (128,80 in a screen size 256x159, facing up.
CLEARSCREEN	CLS	Clear the screen, do HOME and put the pen down.
FORWARD n	FD	Move pen forward n units, where n is an expression (see below)
BACK n	BK	Move pen back n units.
RIGHT n	RT	Turn pen to the right n degrees.
LEFT n	LT	Turn pen to the left n degrees.
REPEAT n [...]	RPT	Repeat everything in square brackets n times; if n omitted, repeat indefinitely.
PENUP	PU	Lift up the pen, so that any movements will not dr a line until a PENDOWN or CLEARSCREEN is done
PENDOWN	PD	Put the pen down.
PENREV	PR	Put pen into reverse mode; it will draw normally, except where it traces over a line, which will be erased.
PENERASE	PE	Pen will leave no trace, and will erase any lines traced over.
EDGE		After an EDGE, the pen will not move past the edge of the screen (this is the normal mode)
NOEDGE		Allow the pen to go past the edge of the screen.
DEFINE name	DEF,TO	Define a procedure with this name; may have zero t three parameters; will delete existing proc. with same name. Finish definition with END.
DELETE name	DEL	Delete this procedure.
LIST		List the names of all procedures.
LIST name		List this one in detail.
SAVE		Save the procedure library to tape
LOAD		Load it back again, overwriting existing procedures

Notes

1. An expression (denoted by n above) may be made up of

   a) An unsigned integer in the range ∅ through 36∅
   or b) the random number generator RND, which will give a result ∅ thru
   or c) the name of a parameter of the proc. you are defining.

   or any sequence of the above connected by +-*/ which will be evaluated left to right. The max. value of an expression is ±32767.

2. To interrupt while the system is drawing, press REPT (not ESC).

Error Messages — Appendix B

No.	Displayed on screen	Explanation
1	MUST START AT LEFT - REINPUT	TO, DEF, DEL, LIST, BYE must all start at the left margin
2	NAME MISSING - RE-INPUT LINE	TO, DEF, DEL must be followed by a name
3		
4	[NEEDED AFTER RPT - RE-INPUT	
5	TOO MANY] - RE-INPUT LINE	)
6	MISSING] - RE-INPUT LINE	) The number of [&] must be equal
7	NOT RECOGNISED - RE-INPUT	Probably a mis-spelt word
8	TOO LARGE (360 MAX)-RE-INPUT	Numbers can only be in the range 0 to 360
9	NOT UNDERSTOOD - RE-INPUT	This word has been recognised, but is not valid here
10	HOW MANY ?? - RE-INPUT LINE	A number is missing after RT,LT,FD etc. or after a name which needs params.
11	ILLEGAL NAME- RE-INPUT	Probably you chose a reserved word
12	NOT RECOGNISED - RE-INPUT	An illegal character
15	ONLY 3 PARAMS ALLOWED	when defining a new name
16	NO ROOM FOR THIS DEFINITION	Delete others to make room
17	CAN'T WHILE DEFINING	Can't do LIST, DEL, BYE, DEF, TO, NOEDG while defining a new word
21	STACK UNDERFLOW - SYS ERR	A serious internal error has occurred
22	TOO MANY PROC CALLS OR RPTS	About 30 nested calls can be made
23	CAN'T FIND THIS NAME	Needs to be defined
25	TOO LARGE NO IN ARITH	32767 is the maximum
26	TOO MANY PARAMS ???	Normally because you have given too many parameters to the proc. just drawn
27	NOT ENOUGH PARAMETERS ???	Normally because this procedure needs more parameters.
28	HIT THE EDGE AND STOPPED	
29	TRYING TO DIVIDE BY ZERO	Can't be done
30	INTERRUPTED	You pressed the REPT or INT key
31	EXPRESSION HAS NEG. VALUE	Illegal on a REPEAT number

Note: errors 1 to 17 are detected when the line is typed in; numbers 21 onwards when the system is drawing.

Turtle Talk Summary of commands

HOME	- Put pen in centre facing up
CLEARSCREEN CLS	- Do Home, put pen down & clear screen
FORWARD n FD n	- Move pen forward n places
BACK n BK n	- Move pen back n places
RIGHT n RT n	- Turn pen to the right n degrees
LEFT n LT n	- Turn pen to the left n degrees
REPEAT n RPT n	- Repeat section in brackets n times (n may be omitted)
DEFINE name DEF name TO name	- Give name (and optional parameters) of procedure to be defined. Existing procedure of this name is deleted.
DELETE name DEL name	- Delete the procedure with this name
LIST name	- Give listing of this procedure. If name omitted, give names of all procs.
PENUP (or PU)	- Lift up the pen from now on
PENDOWN (PD)	- Lower the pen from now on
PENERASE (PE)	- Pen acts as eraser from now on
PENREV (PR)	- Pen acts as "reverser" from now on
NOEDGE	- Allow pen to go over edge without error
EDGE	- Return to normal, with error message
SAVE	- Save procedure library to tape
LOAD	- Load procedure library from tape

To interrupt, press REPT